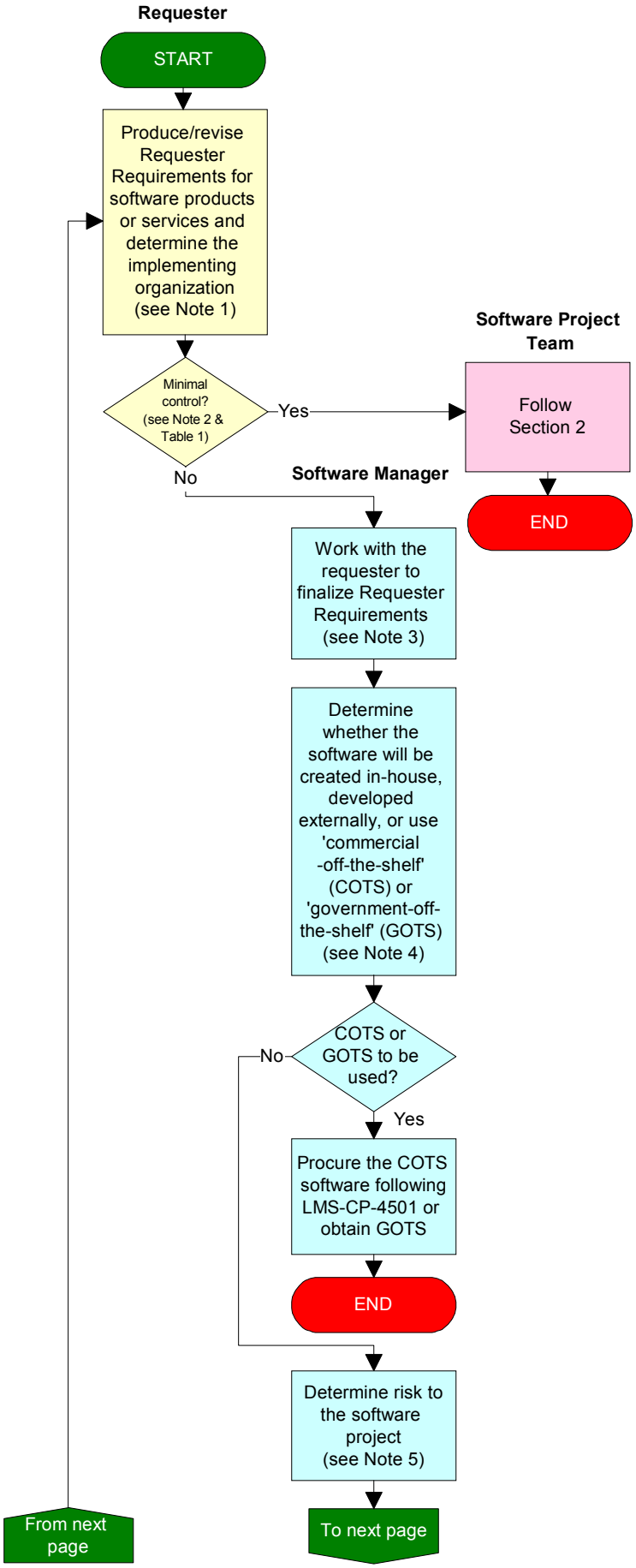
SOFTWARE PLANNING, DEVELOPMENT, ACQUISITION, MAINTENANCE, and OPERATIONS

Objectives:
-to define the process to develop, acquire, maintain and operate software at LaRC
-to ensure requester requirements are met

Approval Original signed by Delma C. Freeman, Jr.
Deputy Center Director

Table of Contents

Section 1: Software Process Flowchart for all Classes of Software



Note 1
This procedure applies to software developed by or for LaRC, including new software and modifications to existing software. Apply this procedure, LMS-CP-5528, if the software or the data produced by the software are delivered or published. The Software Manager must make informed judgments on applying this procedure to the individual project situation.

All major software activities (see definitions) must provide quarterly the metrics specified at URL: http://swmetrics.nasa.gov/

This procedure is for use by all involved in the software process: requesters, supervisors, software managers, software critical-control engineers, and other software project team members. These roles are defined in Appendix A. A single individual can perform multiple roles within a project.

If both the yes and no branches of a decision diamond apply to different parts of a given project, follow both branches.

Organizational Unit Managers and requesters are responsible for integrating software engineering processes with system development and program/project processes.

Note 2
Section 2, Requirements for Applying Minimal-Control to Software, applies to software that:

-Is not a deliverable
-Has negligible risk to LaRC
-Has limited or no maintenance

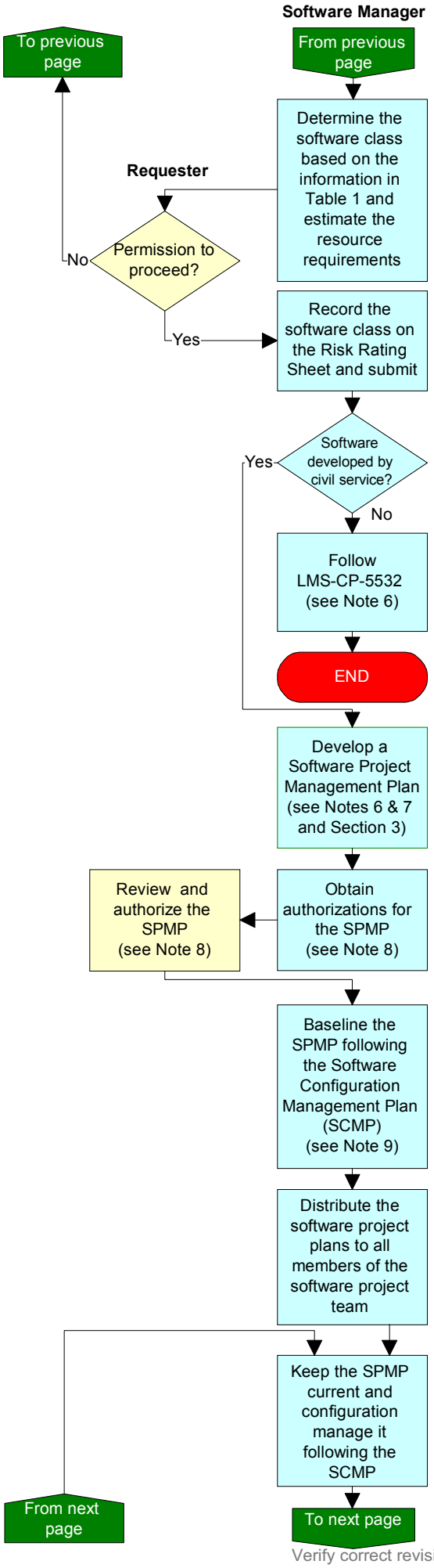
Note 3
It is the shared responsibility of the requesting and implementing organizational units to appoint a Software Manager.

Advice on preparing Requester Requirements is given in the *Software Engineering Guides* [18] see "User Requirements", and guidance on software requirements management is given in the *Capability Maturity Model* [19], section 7.1.

Note 4
To satisfy NASA requirements [11], the Software Manager must determine whether a Trade Study will be performed before software is created or acquired.

Note 5
Initiate an on-line Risk Rating Sheet at URL: http://sw-eng.larc.nasa.gov/process/sheets.html

Each risk area on the Risk Rating Sheet which has a risk rating greater than 2 must be considered for risk management.



Note 6
Software development, maintenance, and operations may be performed by civil servants or contractors. If the work is performed by civil servants, the Software Manager must produce a Software Project Management Plan (SPMP) as described in Section 3. If the work is performed by a contractor, the acquiring Software Manager must produce a Software Acquisition Plan (see LMS-CP-5532), and the contractor must produce an SPMP. If the work is performed by some combination of the above, the acquiring Software Manager must produce both an SPMP (for the parts of the work performed by civil servants) and also a Software Acquisition Plan (for the parts performed under contract). In addition, the contractor must produce an SPMP (for the parts of the work performed by the contractor). For examples of completed plans see [3].

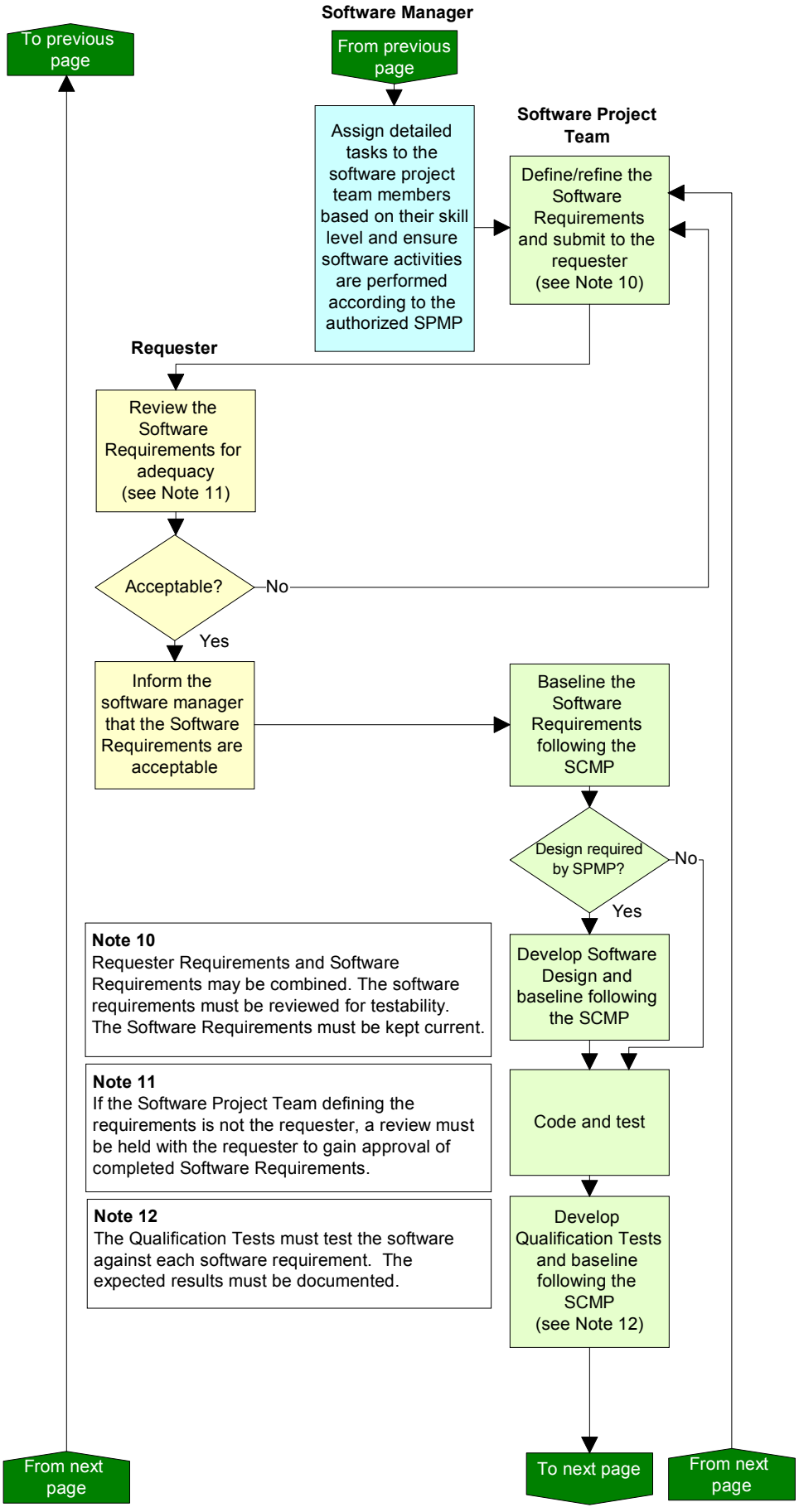
Contracted efforts must be complete and severable tasks.

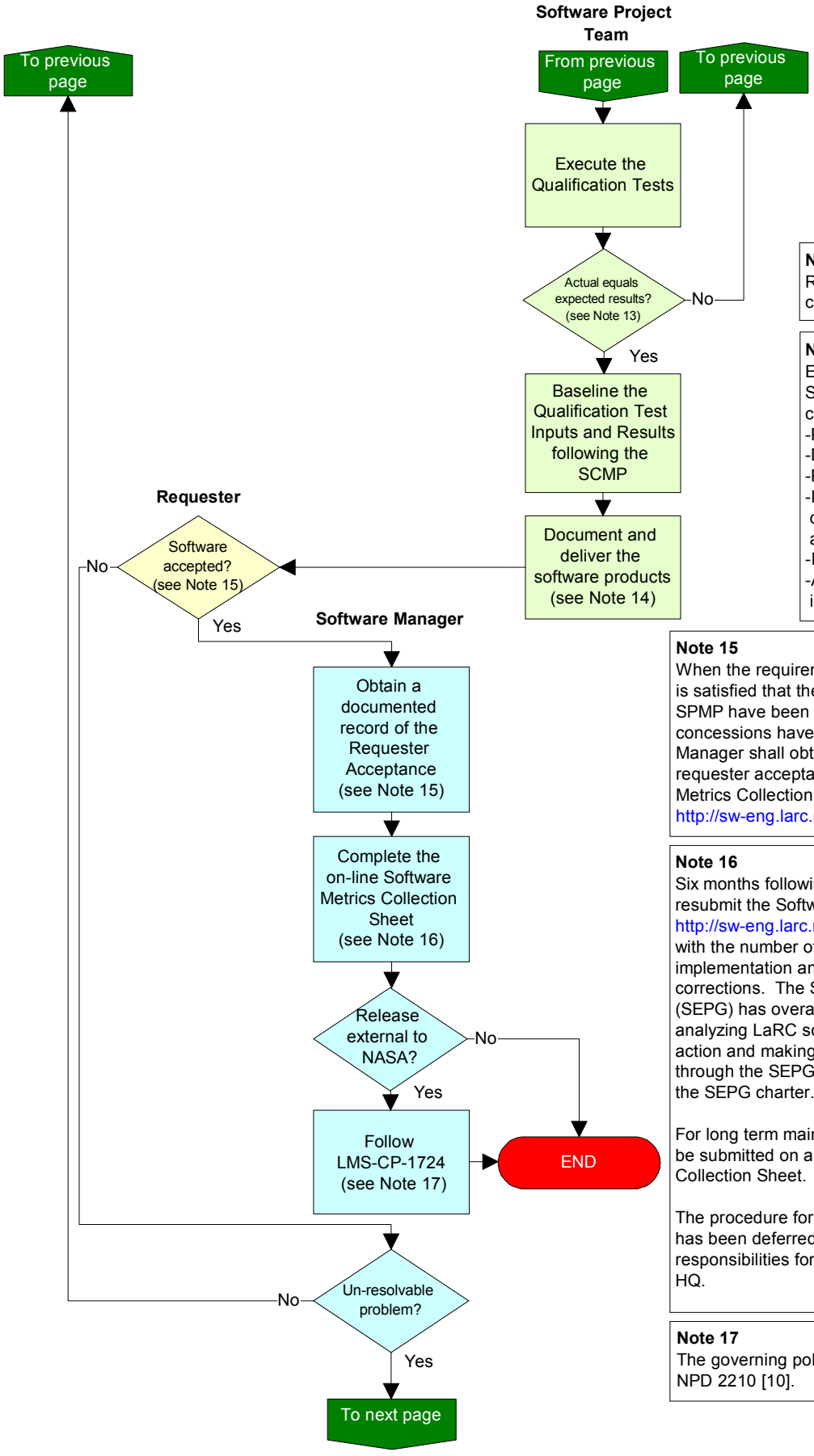
Note 7
For Critical-Control software projects, also follow the requirements given in Section 4.

Note 8
The Supervisor(s) is responsible for authorizing human resources and the requester is responsible for authorizing the work per the SPMP. A record of authorization must be retained.

The Requester must ensure that installation, operation, and maintenance phases of the project are addressed in the SPMP if they are applicable to the project.

Note 9
For High and Critical class software (see Table 1), submit a paper or electronic copy of the Software Project Management Plan to the Head, Office of Mission Assurance (OMA). The software project must not commence until OMA approves High and Critical class SPMPs per LMS-CP-4754.





Note 13
Return to the appropriate activity to make corrections.

Note 14
Each delivery must be documented with a Software Version Description which contains the following information:
-Project title
-Date of delivery
-Point of contact
-Inventory of all baselined CIs to be delivered, including unique CI identifier and description
-Instructions for reading and installing CIs
-A description of all changes incorporated in this delivery

Note 15
When the requirements have been met and the requester is satisfied that the acceptance criteria recorded in the SPMP have been achieved, or when appropriate concessions have been agreed upon, the Software Manager shall obtain a record (e.g. e-mail or signature) of requester acceptance and complete the on-line Software Metrics Collection Sheet at URL:
http://sw-eng.larc.nasa.gov/process/sheets.html

Note 16
Six months following delivery, the Software Manager must resubmit the Software Metrics Collection Sheet at URL:
http://sw-eng.larc.nasa.gov/process/sheets.html
with the number of Problem Reports approved for implementation and the hours spent performing corrections. The Software Engineering Process Group (SEPG) has overall responsibility for collecting and analyzing LaRC software engineering process metrics for action and making them available to the LaRC community through the SEPG web site [3]. LAPG 1150.2 describes the SEPG charter.

For long term maintenance efforts, software metrics must be submitted on an annual basis on the Software Metrics Collection Sheet.

The procedure for collecting metrics in NPD 2820.1 [11] has been deferred until specific metrics and roles and responsibilities for collection are further defined by NASA HQ.

Note 17
The governing policy for external release of software is NPD 2210 [10].

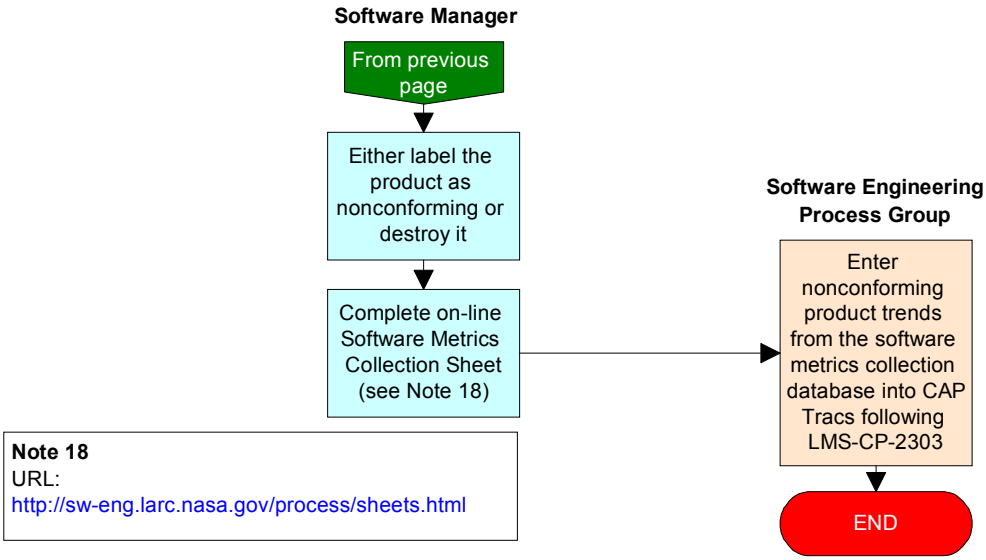


Table 1: Software Classes for Langley Research Center (LaRC)

Software Class	Risk Level	Use of Software or Generated Data	Level of Software Engineering Rigor
Critical Control	Extreme risk to LaRC's reputation, including loss of life or injury/illness to personnel; major damage (in excess of $250K) to equipment, facilities, or the environment; or major collateral damage as a result of tests.	Software which is safety-critical. The common characteristics of the software are that the consequences of failure are so serious that liabilities cannot be fully quantified.	• Critical-control planning and analysis required • Software Project Management Plan (SPMP) and software standards required • Defined development procedures with appropriate analysis of the software before operational use • Formal problem and change management • Formally documented reviews, testing, and maintenance
High Control	Moderate to high degree of risk to LaRC's reputation, including failure of a mission; damage to the Center's reputation or prestige; extended loss of access to a system; or loss of data important to the Center.	Software that may support research or experimentation; may be used in tests; may be used to monitor/operate equipment; may be used to collect, process, model, or simulate data/information or activities; or may require maintenance for the evolution of the software/data.	• SPMP and software standards required • Judicious problem and change management • Documented reviews, testing, and maintenance
Low Control	Small degree of risk to LaRC's reputation.	Software and/or resulting data is delivered to requesters external to the developing organization. Software has limited or no maintenance.	• SPMP is required • Only sufficient rigor to ensure requirements are met and repeatability of results can be achieved • Documented testing
Minimal Control	Has negligible risk to LaRC's reputation.	Software is not a deliverable and has limited or no maintenance.	• Log required • Only sufficient rigor to ensure requirements are met and repeatability of results can be achieved • Documented testing

Section 2: Requirements for Applying Minimal Control to Software

At the point of validation, prior to delivery or publication of the data produced by the software, record the following information in a Log or Research Plan:

- Project title
- Software class
- Version date
- The requirements: software capabilities, outputs, and any constraints on the software
- The tests performed to validate the product(s) and/or data (such as comparison of actual software results with expected lab results and/or results from similar algorithms)

Following validation ensure:

- That all software products, results, corresponding Software Test Inputs and Test Outputs have been given a unique identifier
- That backups of these products have been stored on physically different media
- That the following information has been recorded in the Log:
  - The storage location of the products, results, corresponding Software Test Inputs and Test Outputs
  - How access to these products is controlled
  - Backup/restoration contact and retention period

If removable media is used, ensure the following information is recorded on media labels:

- Project title
- Content description
- Date electronic files were transferred to the media
- Disk or tape sequence number (e.g. Disk 7 of 9)
- Retention period

For software that is operated routinely, describe or reference the activities or tasks which will be followed for software operations.

General Notes:

If "Commercial-Off-The-Shelf" (COTS) or "Government-Off-The-Shelf" (GOTS) will be used to fulfill the requirements, procure COTS products by following LMS-CP-4501.

One Log may be used for multiple projects.

For the user's convenience, the requirements listed above have been posted in Microsoft Word and text templates at URL: http://sw-eng.larc.nasa.gov/process/

Section 3: Software Project Management Plan (SPMP) Requirements for Low, High and Critical Control

3.1 Developing a SPMP

3.1.1 The Software Manager must produce an SPMP. For in-house projects, if software quality assurance support is needed in completing the SPMP, see LMS-CP-4754.

3.1.2 The following software Project Tracking Information from the SPMP and changes to it must be provided to the supervisor:

- Project title
- Software class
- Start and end dates for the work package, or work breakdown structure (WBS) elements, or software lifecycle phases

- Assigned employee names and the percentage of their time per work package, or WBS element, or phase
- Software Manager's name and percentage of time allocated to this role
- Date of approval of the SPMP
- Organization codes when more than one organization is involved in the project

(Note: Document 193 [3] may be used to keep a record of this information.)

3.1.3 The Software Manager must review the requester requirements for completeness, clarity, consistency, and feasibility.

3.1.4 The Software Manager must use the requirements (and changes to the requirements) as the basis for software plans, schedule, work products, and activities. Guidance on software requirements management can be found in *The Capability Maturity Model: Guidelines for Improving the Software Process* [19, section 7.1].

3.1.5 The SPMP must specify or reference the requester acceptance criteria.

3.1.6 The Software Manager must select and document in the SPMP the software life cycle phases that will be used on the project.

3.1.7 The life cycle must include a requester requirements phase and/or a software requirements phase, a code and testing phase, and a qualification-testing phase; in addition, Critical-Control class software projects must also include a design phase. *Guidance on LMS Software Procedures* [1] discusses several life cycle options and contains guidance on choosing one based on the project specifics.

3.1.8 For High-Control and Critical-Control class software projects, the processes, activities, and tasks described in IEEE/EIA 12207.0-1996 [8], must be tailored appropriately to the software project and followed in implementing the software life cycle.

3.1.9 The Software Manager must select and document in the SPMP the software development approach that will be used on the project. *Guidance on LMS Software Procedures* [1] provides recommendations on choosing a development approach based on project size, complexity, and risk.

3.1.10 For Low-Control software projects, the SPMP may take the form of a Log, and it may be for either an individual project or a series of related projects. If more than one project is covered under the plan, the Risk Rating Sheet and the Software Metrics Collection Sheet, must be completed for each project.

3.1.11 For Low-Control software projects, if risks to the project completion are identified, document and track the risk and the associated mitigation or avoidance approach.

Subsections 3.1.12 through 3.1.19 of this procedure do <u>not</u> apply to Low-Control class software.

3.1.12 The SPMP must be developed in accordance with the guidance provided in IEEE/EIA 12207.0 [8], clause 5.2. The SPMP may be a stand-alone plan or included as sections of the project plan. IEEE 1058.1 [5] or similar standard in use (e.g., IEEE/EIA 12207.1, Section 6.11 [9], NASA-STD-2100-91 [12]), may be followed in documenting the SPMP. An outline of the IEEE 1058.1 SPMP is provided in the *Software Project Management Plan Guidance* [2], which also contains guidance on filling out selected sections of the plan.

3.1.13 The minimum contents of software life cycle data documentation must be as described in IEEE/EIA 12207.1-1997 [9], or similar standard in use (e.g., NASA-STD-2100-91 [12]), in conjunction with supplemental documents required for Critical-Control class software. Depending on how the life cycle is tailored, not all life cycle data and documentation will be required, and all life cycle data need not be in the form of separate documents (Refer to IEEE/EIA 12207.1 [9] section 4.3, table 1 for the full listing of life cycle data).

3.1.14 NASA policy [11] requires that — based on the cost, size, life span, and complexity — the plan address "design tradeoff management, risk management [13, Section 4.2][20], requirements

management [19, Section 7.1], software project planning [19, Section 7.2], project tracking and oversight [19, Section 7.3], software product engineering [19, Section 8.5], subcontract management [19, Section 7.4], configuration management [19, Section 7.6], quality assurance [19, Section 7.5], and peer review [19, Section 8.7]".

3.1.15 The Software Manager and project personnel must identify, analyze, plan, track, control and document the risks involved in the software project on a continuous basis. NASA requirements on risk management are provided in section 1.f of NPD 2820.1, and section 4.2 of NPG 7120.5a [13]. The *Continuous Risk Management Guidebook* [20] and the *NASA Continuous Risk Management Course* [17] both offer additional information and guidance on risk management. Note: for the reader's convenience, the 7120.5a requirements on risk management plan contents have been included in the *Software Project Management Plan Guidance* [2] and a risk spreadsheet example is provided at URL: http://sw-eng.larc.nasa.gov/process/ under the "Examples and References" page.

3.1.16 The Software Manager must define the mechanism that specifies how problems will be documented, tracked, and resolved.

3.1.17 The SPMP must specify the procedures to be used for performing the following tracking and oversight activities:

- Authorizing new commitments
- Communicating changes to commitments to software staff
- Tracking and recording actual software size, effort, cost, and schedule of work products against estimates; recording deviations; and recording the revised schedule. Note: When it is vital to the success of the project, computer resource utilization must also be tracked.
- Tracking progress of technical activities and work products, and taking corrective action
- Conducting periodic reviews to track and record technical progress, plans, performance, and issues against the SPMP.

3.1.18 The SPMP must specify the software project tracking and oversight records to be retained.

3.1.19 Documented reviews must be performed according to the SPMP schedule and documented procedure(s).

3.1.20 The results of each review and the names of the reviewers must be documented and retained.

3.1.21 Verification activities must be scheduled. The extent and focus of verification activities will be influenced by the software class and determined by the Software Manager. (See *Guidance on LMS Software Procedures* [1].)

Note: Formal Inspections can be used to satisfy the requirement for software reviews and verification. The use of Formal Inspections is not limited to "source code" in its applicability. Formal Inspections can also be used to find defects in documentation, design products, test, and data. For more information on software Formal Inspections, see the *Instructional Handbook for Formal Inspections* [16].

3.1.22 Validation activities must be scheduled (see *Guidance on LMS Software Procedures* [1]).

Note: Verification, review and validation can be treated as separate activities or integrated and performed as one activity. It is the responsibility of the Software Manager to select the best method for performing these activities.

3.1.23 NASA policy [11] requires the developer to document software "as to its form and function and verify that such software performs the functions claimed on the platform(s) for which it is designed without harm to the systems or the data contained therein."

3.1.24 All software related plans and schedules must be documented and updated to ensure they are current, correct, and feasible. It is recommended that they be reviewed at the end of each phase.

3.1.25 Additional guidance on software planning is found in the *Software Engineering Guides* [18] and the *Software Management Guidebook* [14]. The LaRC Software Process Improvement web site contains examples of products and best practices currently in use at LaRC [3].

3.1.26 The Software Configuration Manager must produce a Software Configuration Management Plan according to the requirements specified in LMS-CP-5529.

3.2 Installation Planning

3.2.1 If installation services are required as part of the software project, the Software Manager must record in the SPMP the mechanisms that will be used for replication, delivery, installation of the software, and training the requester to use the products delivered.

3.2.2 In addition, the plan must define the roles and responsibilities of all involved in the transition process (including the point of contact for requester service).

3.3 Operational Support Planning

3.3.1 If operations support services are required as part of the software project, the Software Manager must record in the SPMP the activities and tasks for which the operator is responsible and the point of contact for requester support.

3.3.2 In addition, IEEE/EIA Standard 12207.0 clause 5.4 [8] must be used for defining the operation process, unless a similar standard is already in use (e.g., NASA-STD-2100-91 [12]).

3.4 Maintenance Planning

3.4.1 If maintenance services are required as part of the software project, the Software Manager must record the following in the SPMP: the level of maintenance to be performed (e.g., modify only to fix problems, or modify to include fixes and enhancements); how problems and/or modifications are identified, classified, prioritized, tracked, and analyzed; and the approval, implementation, and test process to be used.

3.4.2 IEEE/EIA Standard 12207.0, clause 5.5 [8] must be used for defining the maintenance process unless a similar standard is already in use (e.g. NASA-STD-2100-91 [12]). It is recommended that the IEEE Standard 1219-1998 [7] be used for developing the plan.

3.4.3 If the project only involves maintenance, the Maintenance Plan satisfies the requirements for an SPMP.

3.4.4 Where an external contractor performs maintenance, the plan must form the basis for defining the work requirements of the contractual agreement.

3.5 Other Plans

3.5.1 If applicable to the software project, the Software Manager must include a plan to address requirements for health, safety, systems administration, and security (e.g., proprietary, classified, financial, etc.) in the SPMP.

Section 4: Requirements for Applying Critical Control to Software

4.1 Additional Responsibilities

4.1.1 The Software Manager must ensure that a Hazard Analysis has been performed by the Head of the Office of Safety and Facility Assurance.

4.1.2 The Software Critical-Control Engineer must be able to report any critical-control concerns to the Systems Engineer.

4.1.3 The Software Manager is responsible for naming the Software Critical-Control Engineer, the leader of the Software Project Team, and the leader of the Verification and Validation (V&V) Team in the SPMP.

Note: Additional guidance on software safety can be found in [15].

4.2 Software Integrity Levels

4.2.1 The Software Manager must ensure that a Software Integrity Level (SIL) is assigned to each software component which can cause or contribute to a hazard. A SIL is assigned based on hazard(s) identified in the Hazard Analysis (See Table 4-1). Use Table 4-2 to determine the techniques to apply during development.

4.2.2 Software Manager must **not** lower a SIL below that indicated by the Hazard Analysis.

4.2.3 Within a software component, all software units must have the same SIL.

4.2.4 For SILs S3 and S4, the Verification and Validation (V&V) Team must be comprised of different personnel than those of the Software Project Team.

Table 4-1: Mapping of Hazards to Software Integrity Levels

Safety-critical Hazards	SIL
• Loss of life or significant injury/illness to personnel • Major damage to equipment, facilities, or the environment • Major collateral damage as a result of tests	S4
• Injury/illness to personnel • Damage to equipment, facilities, or the environment • Collateral damage as a result of tests	S3
• Moderate endangerment of personnel • Moderate damage to equipment, facilities, or the environment • Moderate collateral damage as a result of tests	S2
• Minor endangerment of personnel • Minor damage to equipment, facilities, or the environment • Minor collateral damage as a result of tests	S1

Table 4-2: Application of Techniques to be Used Based on Software Integrity Level

	Software Integrity Level (SIL)			
Technique	**S4**	**S3**	**S2**	**S1**
Software Specification	semi-formal or formal (if feasible)	semi-formal (rigorous)	informal, natural language	Informal, natural language
Prototyping	yes	yes		
Defined Design Process	yes	yes	yes	yes
Requirements & Design Reviews	by whole project team	by project team	by project team	by peer review
Configuration Management	formal	formal	informal	informal
Review of Hazard Analysis Report	by whole project team	by project team	by project team	by project team
Coding Languages	safe subset of high-level language	safe subset of high-level language	high-level language	
Defensive Programming	yes	yes	yes	
Compiler	validated	validated		
Object Code (vendor-supplied)	verified	verified		
Worst Case Execution Time	analyzed	analyzed		
Static Code Analysis	yes	yes		
Formal Inspection	yes	yes	Yes	yes
Testing Coverage:				
• High-level requirements	yes	yes	Yes	yes
• Low-level requirements	yes	yes	Yes	yes
• Data and statements	yes	yes	Yes	
• Branches	yes	yes		
V&V	different than developer	different than developer	by developer	by developer

Supporting definitions for Table 4-2:

Configuration Management–formal: Characterized by the use of documented procedures for controlling and tracking software changes; tools are used to automate version control and change tracking.

Configuration Management–informal: Characterized by the use of documented procedures for controlling and tracking software changes; personnel manually perform version control and change tracking.

Safe subset of high-level language: Vendor-supplied, nonstandard features are not used or use is minimized and isolated.

Compiler–validated: The full set of language statements to be used are employed to determine allowable spelling deviations, order dependency, and comment recognition and to confirm optional syntax, statement termination requirements, maximum length of variable names, scope of variables, operator precedence, addressing schemes, and other similar issues. Vendor validation of a compiler covering the above items is acceptable.

Defensive programming: Characterized by coding practices which eliminate or minimize the probability that an input will cause an application to fail. Considerations include explicitly declaring and initializing all variables, context-checking all input variables (i.e., matching actual vs. expected type of input), boundary-checking all variables, providing default branch conditions, avoiding inheritance features after initialization, and other similar practices.

Object Code (vendor-supplied)–verified: Vendor-supplied library functions to be used are exercised in a test application to ensure that correct results are obtained. The test application calls library functions to confirm required and optional calling arguments, the type of required and optional calling arguments, side effects of invalid or missing arguments, boundary checks performed on arguments, returned error codes, and other similar issues.

Testing Coverage–data and statements: Data values are examined before and after the execution of a statement to verify that resultant value(s) are computed correctly and that only values which are expected to change are affected.

4.3 Software Critical-Control Plan

4.3.1 The Software Critical-Control Plan is a supplement to the SPMP. The plan must be developed in the initial stages of the project in order to provide visibility of all the activities that will contribute to the assurance of the critical control of the software. The outline for the Software Critical-Control Plan shown in Table 4-3 must be followed.

4.3.2 The Software Critical-Control Engineer must write the Software Critical-Control Plan.

4.3.3 The Software Manager must approve the Software Critical-Control Plan.

4.3.4 The Software Project Team must execute the Software Critical-Control Plan.

4.3.5 The Software Critical-Control Plan must be kept current.

Table 4-3: Software Critical-Control Plan

SOFTWARE CRITICAL-CONTROL PLAN
1. Purpose and Scope State the purpose and scope of the plan, including the critical-control goals that are expected to be achieved.
2. Definitions and References
3. Management Specify any additional specialized procedures and practices above those specified in the Software Project Management Plan. Include information that defines the level of independence of project activities and addresses staff competence to undertake the software project.
4. Procedures and Practices Describe arrangements for coordination on critical-control matters between organizations participating in the development of the total system and define circumstances under which matters relating to Hazard Analysis are referred to these other parties.
5. Disposition of Hazard Analysis Identify the software components for which the software critical-control analysis will be performed and document their respective SILs. If a higher SIL is applied for any software component, it is justified here.

4.4 Software Critical-Control Analysis

4.4.1 The Software Critical-Control Engineer must perform software critical-control analysis throughout the project life cycle. The purpose of the software critical-control analysis is to evaluate potential failures that may cause new hazards or contribute to existing ones and ensure that critical-control features are correctly implemented.

4.4.2 The Software Critical-Control Engineer must document the software critical-control analysis in the Software Critical-Control Analysis Report. The outline for a Software Critical-Control Analysis Report shown in Table 4-4 must be followed.

4.4.3 The Software Critical-Control Analysis Report must be available to the Project Manager for inclusion in the Hazard Analysis updates.

Table 4-4: Software Critical-Control Analysis Report

SOFTWARE CRITICAL-CONTROL ANALYSIS REPORT
1. Description of the system/subsystem(s) Provide a high level description of the system and its corresponding subsystems (e.g. facility safety interlock system, wind tunnel model protection system, etc.).
2. Description of the hardware Identify the number of processors and type, memory capacity, analog input/output boards, digital input/output boards, network interfaces, etc. References to drawings are appropriate.
3. Description of the software Provide a brief operational description of the software for the system.
4. Software Critical-Control Analysis Describe how and why the following items mitigate the potential hazard(s) for the SIL assigned to each software component: a. Software requirements—i.e., analysis, reviews, formal inspection, etc. b. Software design—i.e., method, tools, analysis, reviews, formal inspection, etc. c. Software code—i.e., language, tools, coding practices, static code analysis (e.g., flow, functional, etc.), reviews, formal inspection, etc. d. Testing—i.e., phases of testing, coverage of testing (e.g., paths, statements, branches, etc.), and regression testing performed following correction to defects. If applicable, discuss areas where alternative methods (in lieu of software) mitigate the potential hazard(s) (e.g., operating procedures, limitations, etc.).
5. Summary Summarize and make the claim that the software is suitable for use in the system.

4.5 Software Critical-Control Reviews

4.5.1 The following critical-control items must be covered at reviews specified in the SPMP:

- Hazard Analysis
- Software Critical-Control Plan
- Software Critical-Control Analysis Report

4.5.2 Reviews must include the following panel members: Software Critical-Control Engineer, Leader of the V&V Team, Systems Engineer, Requester Representative, and Office of Safety and Facility Assurance Representative.

Appendix A: Definitions

Baseline: (1) A specification or product that has been formally reviewed and agreed upon that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. (2) A document or a set of such documents formally designated and fixed at a specific time during the life cycle of a configuration item [4].

Commercial-Off-The-Shelf (COTS) software: A general-purpose application, utility or system developed and sold by a company and for which a means of providing through-life support is available.

Computer system: A system containing one or more computers and associated software [4].

Configuration management: A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements [4].

Delivery: Release of a system or component to its requester or intended user.

Government-Off-The-Shelf (GOTS) software: Software developed by the Government.

Hazard Analysis: A component-by-component system evaluation where possible failures are examined to determine the probability of occurrence and resulting consequences.

Life cycle: See Software life cycle.

Log: A software engineer's file, with a unique identifier, containing software information and phase outputs. A Log may be a developer's notebook or electronic file.

Low-control software: Software that has limited or no maintenance. The software and/or resulting data may be delivered to requesters external to the developing organization.

Maintenance: The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [4].

Major software activities: Aerospace programs or projects in which failure of the software could cause mission failure; projects in which failure of the software could result in harm to humans, facilities, or equipment; or projects in which failure of the software would cause risk to NASA's public reputation.

Peer review: A review of a software work product, following defined procedures, by peers of the producers of the product for the purpose of identifying defects and improvements [19].

Phase: A major segment of work in the software development process, for example: requester requirements, software requirements, architectural design, detailed design, coding and testing, qualification-testing, acceptance, operations, and maintenance.

Requester: The person responsible for funding the software project and receiving project deliverables. The IEEE standards use the term "customer" to refer to requester.

Software: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system [4]. Examples include code, code generated using CASE tools, databases, graphical user interfaces, object libraries, mathematical or data analysis packages, and requirements, design, and test documents.

Software Configuration Manager: The individual responsible for development of the Software Configuration Management Plan and ensuring it is executed.

Software critical-control analysis: An activity in which system or software requirements, software design, and software code are examined to identify defects that have a potential to cause or contribute to system hazards.

Software Critical-Control Engineer: A software engineer who applies safety engineering principles to the formulation, design, development, testing, and documentation of software.

Software Engineer: A member of the software technical staff who applies engineering principles to the formulation, design, development, testing, operations, and maintenance of software.

Software Engineering Process Group (SEPG): A group of specialists who facilitate the definition, maintenance, and improvement of the software process used by the organization [19].

Software life cycle: The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. Note: These phases may overlap or be performed iteratively [4].

Software Manager: The individual with overall responsibility for planning, control, and delivery of a software project.

Software metric: 1) A quantitative measure of the degree to which a system component or process possesses a given attribute; or 2) a measurable indication of some quantitative aspect of a software project; (e.g., size, cost, risk, time).

Software project: A number of specified activities encompassing the acquisition, supply, development, operations, or maintenance of software. A software project may be 1) a project in its own right, or 2) a subproject of a parent project.

Software Project Management Plan (SPMP): A document which covers the totality of a software project from start to finish and describes the objectives and deliverables, the approach to be taken, the controls employed, the activities and milestones, and the resources to be used.

Software Project Team: The personnel assigned to a project, who design, code, unit test, and document software.

Supervisor: An individual whom higher management has given responsibility and authority for assigning workforce, managing facilities, and reviewing work plans for technical accuracy and validity.

Validation: The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements [4].

Verification: 1) The process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase. 2) Formal proof of program correctness [4].

V&V Team: The personnel assigned to a project who review testing of software and prepare and conduct plans to determine that software operates correctly (i.e., verify) and operates according to requirements (i.e., validate).

Appendix B: References

1 Guidance on LMS Software Procedures. (URL: http://sw-eng.larc.nasa.gov/process/)

2 Software Project Management Plan Guidance. (URL: http://sw-eng.larc.nasa.gov/process/)

3 LaRC Software Process Improvement Initiative web site. (Contains information on the LaRC software practices, contacts, discussion groups, etc.) (URL: http://sw-eng.larc.nasa.gov/)

4 IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

5 IEEE Standard 1058.1-1997, IEEE Standard for Software Project Management Plans.

6 IEEE Standard 1062-1993, Recommended Practice for Software Acquisition.

7 IEEE Standard 1219-1998, IEEE Standard for Software Maintenance.

8 IEEE/EIA Standard 12207.0-1996, IEEE/EIA Standard, Industry Implementation of International Standard ISO/IEC 12207: 1995, Standard for Information Technology—Software Life Cycle Processes.

9 IEEE/EIA Standard 12207.1-1997, IEEE/EIA Standard, Industry Implementation of International Standard ISO/IEC 12207: 1995, Standard for Information Technology—Software Life Cycle Processes—Life Cycle Data.

10 NASA Policy Directive 2210, October 1997, External Release of NASA Software. (URL: http://nodis.hq.nasa.gov/Library/Directives/NASA-WIDE/Policies/Legal_Policies/N_PD_2210_1.html)

11 NASA Policy Directive 2820.1, May 1998, NASA Software Policies. (URL: http://nodis.hq.nasa.gov/Library/Directives/NASA-WIDE/Policies/Legal_Policies/N_PD_2820_1.html)

12 NASA-STD-2100-91 NASA Software Documentation Standard, 1991. (URL http://satc.gsfc.nasa.gov/assure/docstd.html)

13 NASA Procedures and Guidelines 7120.5a, NASA Program and Project Management Processes and Requirements, April 3, 1998. (See URL: http://nodis.hq.nasa.gov/Library/Directives/NASA-WIDE/Procedures/Program_Formulation/N_PG_7120_5A.html)

14 NASA-GB-001-96, Software Program, Software Management Guidebook. (URL: http://www.ivv.nasa.gov/SWG/resources/index.html)

15 NASA-GB-1740.13-96, NASA Guidebook for Safety Critical Software — Analysis and Development. (URL: http://www.ivv.nasa.gov/SWG/resources/SWG_safety.html)

16 Instructional Handbook for Formal Inspections. (URL: http://sw-eng.larc.nasa.gov/process/)

17 NASA Continuous Risk Management Course taught by the Software Assurance Technology Center, NASA Goddard Space Flight Center, NASA-GSFC-SATC-98-001.

18 *Software Engineering Guides*, C Mazza et al., Prentice Hall (1996), ISBN 0-13-449281-1.

19 Software Engineering Institute at Carnegie Mellon University, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley (1994), ISBN 0-201-54664-7.

20 Software Engineering Institute at Carnegie Mellon University, *Continuous Risk Management Guidebook*, 1996, NTIS#: AD-A319533KKG, DTIC#: AD-A319 533\6\XAB.

Note: The IEEE references listed in this appendix can be obtained through URL: http://sw-eng.larc.nasa.gov/larconly/ieee.html